Solución Dungeons and Dragons

José Santiago Vales Mena

Agosto 14, 2023

Solución

Imagina que tenemos solo un duende mágico en alguna parte del castillo. Ese duende tiene un máximo para derrotar A_i dragones. ¿Cómo podemos resolver ese caso? Justamente, el duende nos dice que antes de él debemos tomar los A_i números con mayor valor, y todo lo que está después de él lo tomamos sin problemas.

Ahora extendamos esa idea cuando tenemos más duendes en el camino. Tengo un conjunto de dragones que puedo derrotar hasta encontrar el primer duende. Cuando encuentro el primer duende, este me indica que solo puedo derrotar D_1 de esos dragones. Entonces, descarto los dragones que me aportan menos oro hasta tener una cantidad de D_1 dragones. Luego, consideramos que podemos derrotar a todos los dragones que vienen después de este duende. Sin embargo, cuando aparece el segundo duende, nos dice que podemos derrotar hasta D_2 dragones. Nuevamente, aplicamos el proceso de descartar dragones hasta tener D_2 dragones, y así sucesivamente.

Por lo tanto, necesitamos ser capaces de eliminar los dragones de menor costo hasta quedarnos con la cantidad deseada de forma eficiente. Para eso, podemos usar una estructura llamada **Heap**. Metemos los dragones en la *Heap* y cada vez que aparece un duende, vaciamos la heap hasta quedarnos solo con los D_i valores más grandes. Para esto, necesitamos usar una *min heap* ya que eliminaremos los valores menores. Además, dado que cada elemento solo se inserta y se elimina una vez, tenemos una complejidad total de $O(N \log N)$, lo cual es suficiente para resolver el problema.

Recuerden que podemos lograr una min heap insertando los números negativos en la ***priority_queue*** *de la librería.*

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N;
    cin >> N;
    vector<char> personajes(N);
    vector<int64_t> vals(N);

    for (auto& it : personajes) cin >> it;
```

```
    for (auto& it : vals) cin >> it;

    priority_queue<int64_t> tomar;
    for (int i = 0; i < N; i++) {
        if (personajes[i] == 'D') tomar.push(-vals[i]);
        else {
            while (tomar.size() > vals[i]) tomar.pop();
        }
    }

    int64_t suma = 0;
    while (!tomar.empty()) {
        suma -= tomar.top();
        tomar.pop();
    }

    cout << suma << '\n';
    return 0;
}
```